(19)日本国特許庁（ＪＰ）　(12)特　許　公　報（Ｂ２）　(11)特許出願公告番号

特公平6－19731

(24)(44)公告日　平成６年(1994)３月16日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　11/28	３４０　Ａ	9290－5Ｂ			
9/45					
		9292－5Ｂ	Ｇ０６Ｆ　9/44	３２２　Ｃ	

発明の数１(全　８　頁)

(21)出願番号　特願昭59－250174

(22)出願日　昭和59年(1984)11月27日

(65)公開番号　特開昭61－127044

(43)公開日　昭和61年(1986)６月14日

審判番号　平4－10826

(71)出願人　999999999
富士通株式会社
神奈川県川崎市中原区上小田中1015番地

(72)発明者　柳沢　実
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(74)代理人　弁理士　山谷　晧榮

審判の合議体
審判長　松尾　浩太郎
審判官　大橋　隆夫
審判官　吉岡　浩

(56)参考文献　特開　昭58－175061（ＪＰ，Ａ）
「新版　情報処理ハンドブック」情報処理学会編　昭和55年３月30日　オーム社発行　Ｐ．252－260

(54)【発明の名称】　プログラムの単体テスト方式

【特許請求の範囲】

【請求項１】複数のモジュールが階層的に構成されるプログラムのモジュール単体を、プロセッサが実行してテストを行うプログラムの単体テスト方式において、

該被テストモジュールを解析して、上位及び下位モジュールの存在を検出して、上位又は下位のモジュールの代替機能を行うテスト支援ツールを設け、

該テスト支援ツールは、該上位モジュールの存在を検出した場合には、該被テストモジュールに渡すパラメータの領域をメモリに確保して、テストデータをセットした後、該被テストモジュールを起動せしめ、

該下位モジュールの存在を検出した場合には、該メモリにパラメータ領域を得て、該被テストモジュールのテスト実行中に、該下位モジュールが呼び出されたことに応じて、該被テストモジュールに渡すテストデータを該パラメータ領域にセットして、該被テストモジュールに渡すことを特徴とするプログラムの単体テスト方式。

【発明の詳細な説明】

〔産業上の利用分野〕

本発明は，複数のモジュールで構成されるプログラムの各モジュールプログラムを単体でテストするプログラムの単体テスト方式に関し，特に関連する他のモジュールが作成されていなくてもテストできるプログラムの単体テスト方式に関する。

一般に大きなシステムにおいては，プログラムが１本のモジュール（コンパイル単位）で構成されているものは少なく複数のモジュールが集まって一つの処理を行なうように階層的に構成されている。

例えば，第５図に示すようにモジュールＡのサブルーチンとしてモジュールＢが，モジュールＢのサブルーチン

としてモジュールＣ，Ｄが構成されて１つのプログラムを構成するようにしている。このような階層的なプログラムを開発するには，テストの効率を考慮してモジュールの作成順序等が決定され，一般に第６図で示す如くシステム設計された後，各モジュールＡ，Ｂ，Ｃ，Ｄが作成される。

この開発の手順としては，上位のモジュールＡから作成するトップダウン方式と，下位のモジュールＣ又はＤから作成するボトムアップ方式とがある。

〔従来の技術〕

このような各モジュールが階層的に構成されたプログラムをテストするには，全てのモジュールが完成してからプログラムをテストすることが望ましいが，各モジュールが順次作成されていくような場合には，各モジュールの作成毎にそのモジュールを単体でテストするほうが開発手順上効率がよい。

しかしながら，モジュールを単体でテストするにしても，階層的なプログラムの構成では，関連する他のモジュールが存在しないとテストできない。例えばモジュールＡをテストするときは，モジュールＢが必要であり，モジュールＢをテストするときは，モジュールＡ，Ｃ，Ｄが必要となる。このため，従来は，テストするモジュールに関連する仮モジュールをテスト担当者が作成し，係るモジュールの単体テストを行っていた。

例えば，モジュールＢをテストするため上位のモジュールＡの仮モジュール（ドライバという）及び下位のモジュールＣ，Ｄの仮モジュール（スタブという）を作成していた。

〔発明が解決しようとする問題点〕

このようなモジュールのテストで必要となるドライバやスタブは仮モジュールなので小さく簡単に作成できるが，通常のプログラム作成と同様にコーデイングやこれのテストの工数を要し，単体テストのためのドライバやスタブの開発をモジュール個々に要するという問題があった。

これを解決するため、被テストモジュールの下位モジュール代替機能を行うスタブ機能を設けたテスト方法が提案されている（例えば、特開昭５８－１７５０６１号公報等）。

このようにしても、次の問題があった。

①上位モジュールの代替機能がないため、トップダウンプログラミングしか適用できず、ボトムアッププログラミングや両者の混在における単体テストができない。

②下位モジュールの代替時に、使用領域等を設定する必要があり、データ設定に手間がかかる。

〔問題点を解決するための手段〕

従って、本発明は、被テストモジュールを解析して、上位、下位モジュールの存在を検出して、上位、下位の代替機能を自動設定することができるプログラムの単体テスト方式を提供することを目的とする。」

このため、本発明は、複数のモジュールが階層的に構成されるプログラムのモジュール単体を、プロセッサが実行してテストを行うプログラムの単体テスト方式において、該被テストモジュールを解析して、上位及び下位モジュールの存在を検出して、上位又は下位のモジュールの代替機能を行うテスト支援ツールを設け、該テスト支援ツールは、該上位モジュールの存在を検出した場合には、該被テストモジュールに渡すパラメータの領域をメモリに確保して、テストデータをセットした後、該被テストモジュールを起動せしめ、該下位モジュールの存在を検出した場合には、該メモリにパラメータ領域を得て、該被テストモジュールのテスト実行中に、該下位モジュールが呼び出されたことに応じて、該被テストモジュールに渡すテストデータを該パラメータ領域にセットして、該被テストモジュールに渡すことを特徴とする。

〔作用〕

本発明では，テスト支援ツール（プログラム）が設けられるが，この支援ツールがドライバ又はスタブ機能を自動的に実行するようにしている。即ち，支援ツールは被テストモジュールを解析し，これに含まれる上位又は下位モジュールのインターフェイス（接続）情報を抽出し，上位又は下位モジュールの存在を検出し，上位モジュールが存在する時は被テストモジュールの起動に先立ち，インターフェイス情報を基にパラメータ領域をメモリに確保し，更にこの領域に外部から与えられたテストデータをセットして被テストモジュールのテスト起動を行ない，又下位モジュールが存在するときは，該被テストモジュールのテスト実行中に下位モジュールを呼出すことに応じてインターフェイス情報から外部から与えられたテストデータをメモリにセットして被テストモジュールに渡すようにして被テストモジュールのテストを続行させるようにしている。

〔実施例〕

以下，本発明を実施例により詳細に説明する。

第１図は本発明の一実施例ブロック図であり，図中，１はコンピュータ（プロセッサ）であり，プログラムを実行して所望の処理を行なうもの，１０はそのＯＳ（オペレーティングシステム）部であり，コンピュータ１全体の制御を行なうもの，１１はモジュール解析部であり，後述する如く論理設計言語で記述された論理設計情報を解析してＣＯＢＯＬのソースプログラムを作成し，且つインターフェイス（デバッグ）情報を抽出するもの，１２はＣＯＢＯＬコンパイラであり，ソースプログラムをコンピュータの実行できる形式のロードモジュールに変換するもの，１３はテスト支援ツール部であり，後述する如くインターフェイス情報からドライバ／スタブ機能を行なうものである。尚，ＯＳ部１０，モジュール解析部１１，ＣＯＢＯＬコンパイラ１２及びテスト支援ツール部１３はコンピュータ１の実行する機能をブロック化したものである。２はファイルメモリであり，各種ファ

イルを格納しておくもの，２０は論理設計情報ファイルであり，プログラマーが作成した後述する論理設計情報を格納しておくもの，２１はＣＯＢＯＬソースファイルであり，モジュール解析部１１によって作成されたＣＯＢＯＬソースプログラムを格納しておくもの，２２はデバック情報ファイルであり，モジュール解析部１１で抽出されたインターフェイス情報を格納しておくもの，２３はロードモジュールファイルであり，ＣＯＢＯＬコンパイラ１２が作成したロードモジュールを格納しておくもの，２４はテストデータファイルであり，オペレータが入力したテストデータを格納しておくもの，３はコンソールであり，ディスプレイとキーボードから成り，オペレータが操作して論理設計情報やテストデータを入力するものである。

次に、第１図実施例構成の動作について第２図の処理フロー図，第３図の論理設計情報説明図及び第４図のテスト支援ツールの動作説明図を用いて説明する。

以下，周知の高級言語である論理設計言語によりプログラミングされたものについてその動作を説明する。

①論理設計言語によるプログラミングは，第３図に示す如く各モジュールＡ，Ｂ，Ｃ，Ｄで共通なフオーマット定義書（第３図(A)），各モジュールのインターフェイス定義書（第３図(B)），各モジュールのモジュール定義書（第３図(C)）等を作成することにより行なう。この共通フオーマット定義書は各モジュールで共通な項目（データ）の形式，桁数等の定義を行なうものであり，インターフェイス定義書は各モジュールの入出力項目，モジュール名等を定義するためのものであり，基本設計段階で決定され，コンソール３よりコンピュータ１を介しファイルメモリ２の論理設計情報ファイルに格納されている。

一方，モジュール定義書は各モジュール毎に作成され，当該モジュールの処理や論理，その条件が手順に従って羅列されたものであり，プログラマーは各モジュールに必要な処理等を順次記述してプログラミングし，これが当該モジュールの処理内容を示す。オペレータはこのモジュール定義書を完成する毎にコンソール３からコンピュータ１を介し論理設計情報ファイル２０に格納せしめる。

例えば，モジュールＢのモジュール定義書が完成し，論理設計情報ファイル２０に格納され，他のモジュールＡ，Ｃ，Ｄは完成していないとする。

②オペレータはコンソールよりコンピュータ１にＯＳ部１０のジヨブ制御機能を働かせ，モジュールＢのＣＯＢＯＬソースプログラムを作成するように指令する。これによってコンパイラの一種であるモジュール解析部１１は論理設計情報ファイル２０の論理設計情報を読出し，ＣＯＢＯＬで記述されたソースプログラムを作成し，ＣＯＢＯＬソースファイル２１に格納する。このＣＯＢＯＬで記述されたソースプログラムは，例えば第４図の被テストモジュールとして示すものであり，論理設計言語を用いずプログラマーがＣＯＢＯＬでプログラミングしたものに相当する。

③これとともにモジュール解析部１１は論理設計情報からインターフェイス情報を抽出する。例えば，第３図の例では，モジュールＢはモジュールＡに呼び出され，パラメータの数はＧ１，Ｇ２の２ツでその領域はＡ１，Ａ２であることがモジュールＢのインターフェイス定義書より抽出され，そのパラメータＧ１，Ｇ２の長さは共通フオーマット定義書より抽出される。又，モジュールＢのモジュール定義書よりモジュールＣ（日数＝%日数計算），モジュールＤを呼び出すことが抽出され，その使用領域はモジュールＣ，Ｄのインターフェイス定義書より抽出される。

これをＣＯＢＯＬソースプログラムの記述から見ると，第４図の被テストモジュールの記述の如く，モジュールＡのＡ１，Ａ２を用い，モジュールＣのＢ１，モジュールＤのＢ２を用いることになる。

モジュール解析部１１はモジュールＢが呼び出されるときのインターフェイス（モジュール名，パラメータの数，パラメータの領域名，その長さ等）及びモジュールＢが呼び出す時のインターフェイスを論理設計情報から抽出し，デバッグ情報ファイル２２に格納する。

このようにしてモジュールＢに対するＣＯＢＯＬソースプログラムが作成され，且つモジュール間のインターフェイス情報が抽出される。

④次に，モジュールＢの単体テストを行なう。

このため，オペレータはコンソール３よりコンピュータ１を介しモジュールＢのテストのためのテストデータをテストデータファイル２４に格納する。そして，オペレータはテスト実行指令をコンピュータ１に入力する。コンピュータ１では，ＯＳ部１０がＣＯＢＯＬコンパイラ１２を動作せしめ，ＣＯＢＯＬソースファイル２１のモジュールＢのＣＯＢＯＬソースプログラムをコンピュータ１の実行できるロードモジュールに翻訳してロードモジュールファイル２３に格納する。次にＯＳ部１０はテスト支援ツール部１３を起動し，デバッグ情報ファイル２２のインターフェイス情報を解析せしめる。

⑤テスト支援ツール部１３は，第２図の処理フロー図の如く前述のインターフェイス情報を解析し，上位モジュールが存在するかを検出する。上位モジュールが存在するとドライバ機能，即ち上位モジュール代替機能を実行する。先づ，被テストモジュールＢに渡すパラメータの受け渡し領域をコンピュータ１の図示しない内部メモリに確保する。前述の例では，第４図に示す如く内部メモリにＡ１及びＡ２の入出力領域を確保し，そのアドレスをセットしておく。

次に，コンソール３よりオペレータにテストデータの指示を表示し，オペレータはテストデータファイル２４のどのテストデータをセットするかをコンソール３より指

示する。これによって内部メモリのＡ１の領域にテストデータファイル２４の必要なテストデータ（上位モジュールからの入力データ）がセットされることになる。
これによってモジュールＢはパラメータＡ１を用いて処理を実行できる。尚，Ａ２はモジュールＢの実行結果の出力のためのものである。
⑥次に，テスト支援ツール部１３はＯＳ部１０に対し被テストモジュールの起動を指示する。
ＯＳ部１０はロードモジュールファイル２３のモジュールＢのロードモジュールを実行し，その中でモジュールＡのパラメータＡ１を用いて行なう。
⑦このようにしてモジュールＢのテストを実行していくうちに，モジュールＣ又はＤの“ｃａｌｌ”（呼び出し）命令があり，ＯＳ部１０がロードモジュールファイル２３を探索した結果モジュールＣ，Ｄのロードモジュールが見付からなかつたとすると，エラーをテスト支援ツール部１３に通知する。
⑧これによってテスト支援ツール部１３は，被テストモジュールが下位モジュールを呼ぶために用意した内部メモリのパラメータ領域（Ｂ１やＢ２）のアドレスを得，コンソール３よりオペレータにテストデータの指示を表示する。オペレータはテストデータファイル２４のどのテストデータをセットするかをコンソール３より指示する。これによって内部メモリのパラメータ領域にテストデータファイル２４の必要なテストデータ（下位モジュールの出力データ）がセットされることになる。
従って被テストモジュールＢはこのテストデータを下位モジュールの出力データとして受け取り，更にテストを続行していく。
このようにして，テスト支援ツール部１３が，インターフェイス情報から上位又は下位モジュールの代替機能を自動的に実行し，被テストモジュールのテストを行なわしめる。
上述の実施例では，論理設計言語でプログラムした例について説明したが，ＣＯＢＯＬによってプログラムしてソースプログラムを作成してもよく，この場合ＣＯＢＯＬソースプログラムからインターフェイス情報が抽出される。又，テストデータをパラメータ領域にセットするのにコンソール３よりオペレータが指示しているが，これを自動的に行なってもよい。更にソースプログラムの言語もＣＯＢＯＬに限らず，他の言語を用いてもよい。
以上本発明を一実施例により説明したが，本発明は本発明の主旨に従い種々の変形が可能であり，本発明からこれらを排除するものではない。

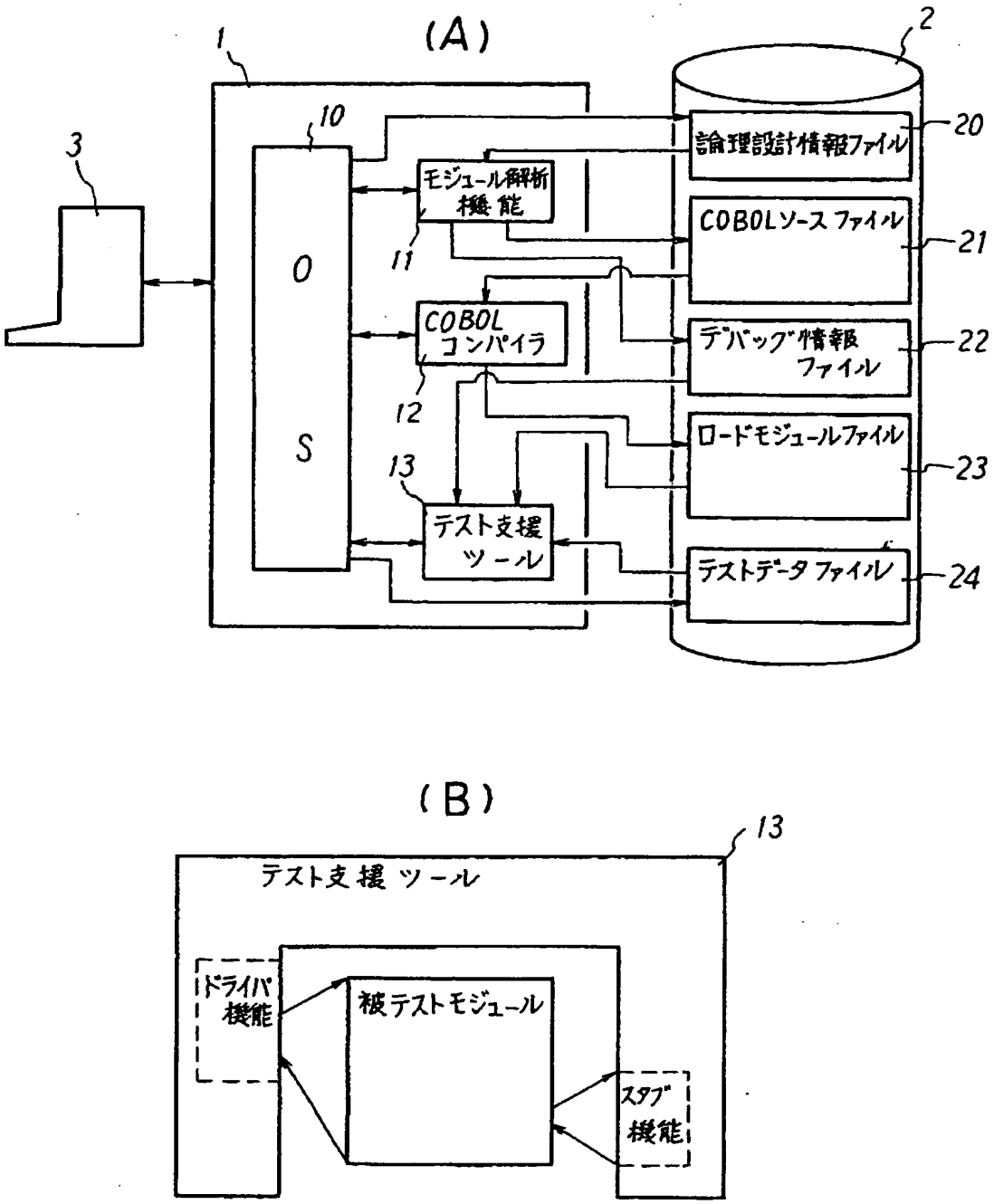
〔発明の効果〕

以上説明した様に、本発明によれば、次の効果を奏する。
①上位及び下位モジュールの存在を検出して、上位又は下位のモジュールの代替機能を行うテスト支援ツールを設けたので、トップダウンプログラミング、ボトムアッププログラミングのいずれでも、単体テストが可能となる。
②被テストモジュールを自動解析して、上位及び下位モジュールの存在を検出して代替機能を行うので、自動的に上位、下位の代替動作が実行できる。
③上位モジュールの存在を検出した場合には、被テストモジュールに渡すパラメータの領域をメモリに確保して、テストデータをセットした後、被テストモジュールを起動せしめるので、上位モジュールの代替機能を自動的に実行できる。
④下位モジュールの存在を検出した場合には、メモリにパラメータ領域を得るので、データ設定が容易となり、単体テストを高速に実行できる。

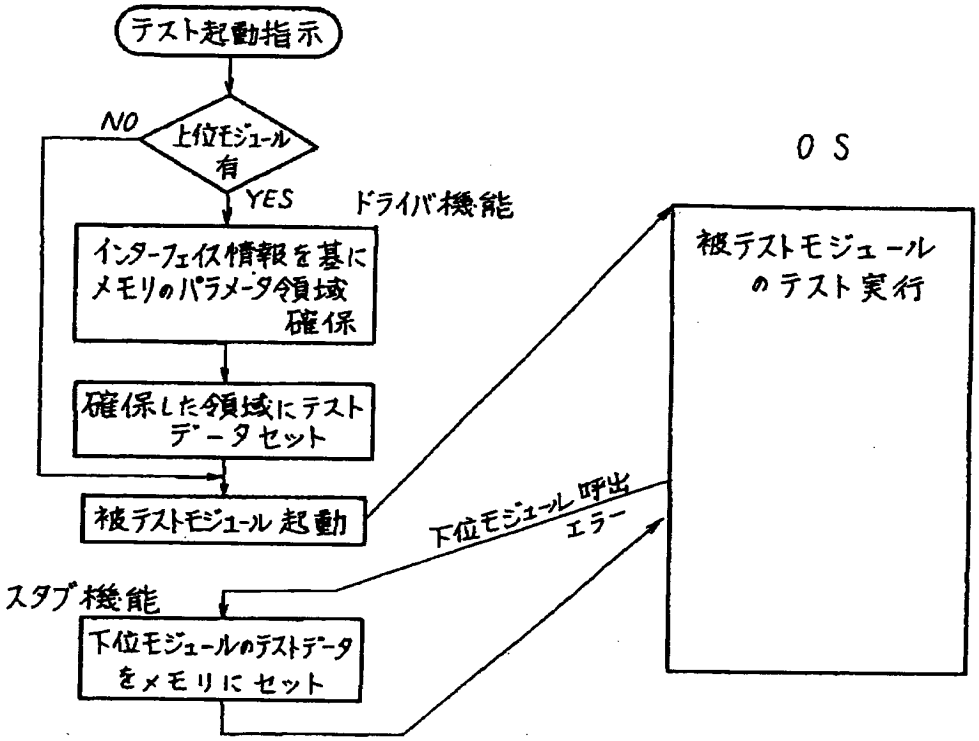
【図面の簡単な説明】

第１図は本発明の一実施例ブロック図，第２図は第１図実施例構成のテスト支援ツールの処理フロー図，第３図は第１図実施例構成に用いられる論理設計言語の説明図，第４図は第１図実施例構成のテスト支援ツールの動作説明図，第５図は階層的モジュール構成プログラムの説明図，第６図は第５図構成のプログラムの開発手順説明図である。
図中，１……コンピュータ（プロセッサ），２……ファイルメモリ，３……コンソール，１３……テスト支援ツール。

【第１図】



【第２図】



【第３図】

(A)

共通フォーマット定数

項番	区分	項　目	形式	桁
10		口座番号	9(4)	
20		科目コード	9(4)	
30		氏　名	N(6)	

(B)

インターフェイス 定義　(モジュールB)

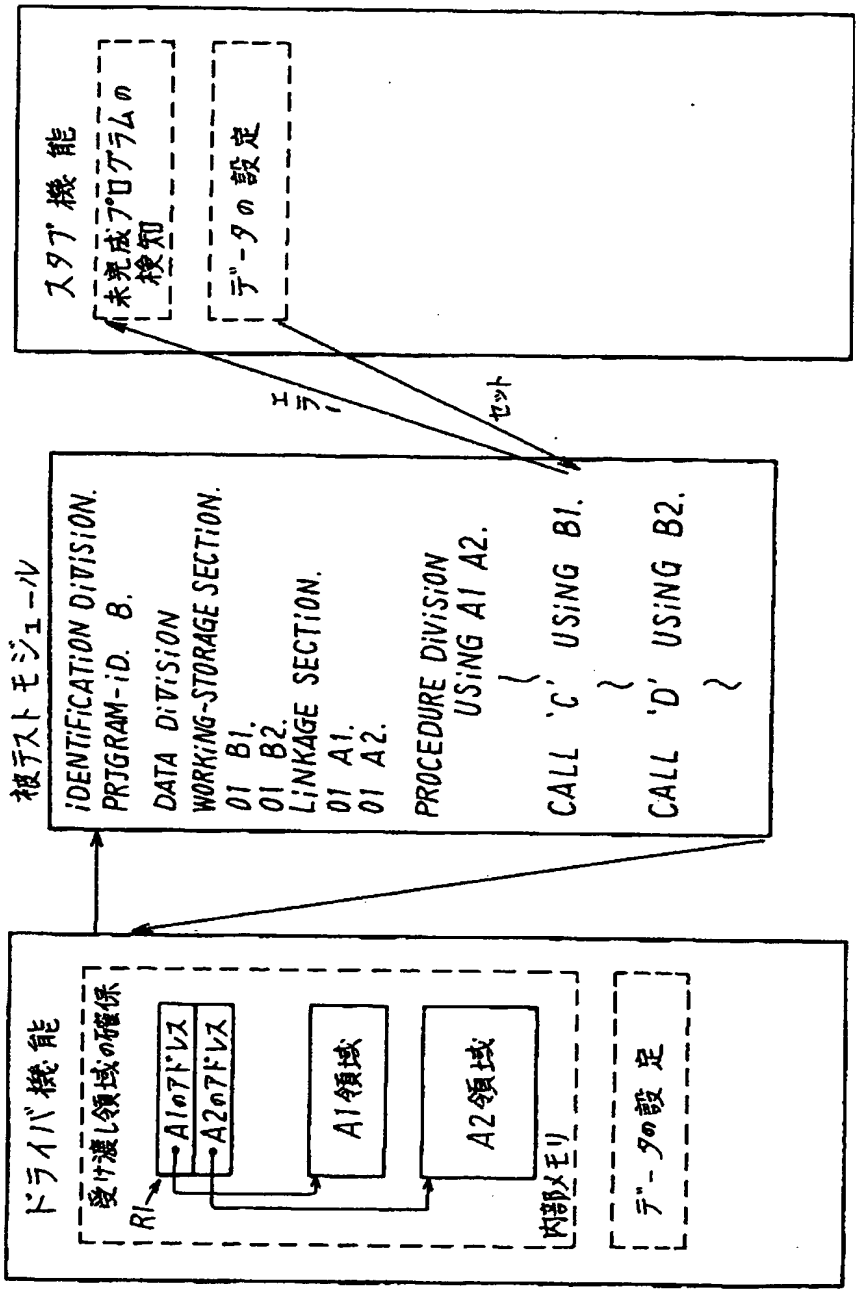
項番	区分	エントリモジュール	入出力項目
1	S	マル優管理 B	GRP:G1 = A1 G2 = A2

(C)

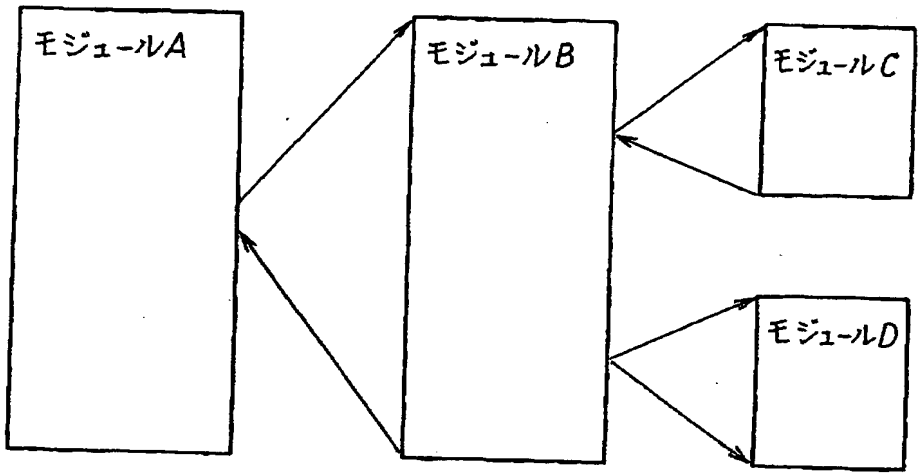
モジュール定義　(モジュールB)

項番	区分	処　理	論理	条件
10				
20				
30		日数＝%日数計算 C		

【第４図】



【第５図】



【第６図】